www.icgst.com

Enhancing Grid Local Outlier Factor Algorithm for Better Outlier Detection

Eslam Mahmoud[1], Ahmed M. Elmogy[1,2], Amany Sarhan[1]
1. *Computers and Control Eng. Dept., Tanta Univ., Egypt*
2. *On leave to Arab East Colleges, Riyadh, KSA*
imhassan@imamu.edu.sa, Amelmogy@arabeast.edu.sa, amany.sarhan@f-eng.tanta.edu.eg

Abstract
Detecting outliers in a large data set is a major data mining task. The existing approaches in this field are categorized into two main categories which are distance-based and density-based outlier detection approaches. Although, Local Outlier Factor (LOF) is considered as the most popular density-based algorithm, it still has some problems related to the speed and accuracy. Enhancing LOF algorithm has been the focus of many researchers working in this field. Among the improved versions of LOF, GridLOF has been proven to have a very good performance. This paper presents an enhancement to GridLOF algorithm by replacing one of its steps by a less complex step which reduces the complexity to be only $O(N)$ instead of $O(N^2)$ in a novel way. The simulation results show that the proposed algorithm outperforms GridLOF algorithm in terms of speed and accuracy.

Keywords: *Outlier, outlier detection, data mining, LOF, GridLOF.*

Nomenclature

LOF	Local Outlier Factor
LOCI	Local Correlation Integral
KLOF	Kernel Density-Based Local Outlier Factor
PNSR	Peak signal-to-noise ratio

1.INTRODUCTION
Big data is a term used to describe the exponential growth of data, both structured and unstructured. In this case the data is so large or complex that traditional data processing techniques are inadequate. Big Data is mostly known to have three characteristics [1]: volume, variety, and velocity. Many factors contribute to the increase in data volume. These factors include, transaction-based data stored through years, unstructured data from social media, and increasing amounts of sensor and machine-to machine being collected. Velocity means that data must be dealt in a timely manner. As technology is evolving very rapidly, dealing with torrents of data in the real time becomes a challenge for most organizations. Finally, as data comes in all types of formats, managing different varieties of data is very important to consider while dealing with big data.

In order to discover patterns, unknown correlations, and other useful information in large data sets, big data analytics are employed [2]. Data mining is the most popular technique for data analytics [3]. This can be used in many areas such as sales, marketing, finance, medicine, supply chain management, and manufacturing. The process of mining patterns from data needs some primary operations. These processes include Data preparation, cleaning, and cleansing [4]. These operations deal with detecting and removing errors and inconsistencies (outliers) from data in order to improve the quality of data. While integrating multiple data sources in big data repositories such warehouses, the need for data cleaning increases significantly. Important issues about data preparation are discussed in [5].

An outlier is defined in [6] as an observation that deviates so much from other observations. Also, in [7], outlier is defined as an observation which appears to be inconsistent with the other members of the same data set. In [8], the terms outlining observation and outlier are used synonymously and are defined as a data observation that appears to deviate markedly from other members in the data set it occurs.

Outlier detection is the process of finding data objects with behaviors that are very different from expectation (outliers) [9]. Outlier detection methods are categorized into three types [3, 9, 10]: statistical methods, proximity-based methods, and clustering-based methods. The proposed work in this paper is some kind of proximity-based method. There are two types of proximity-based outlier detection methods: distance-based and density-based methods. A distance-based outlier detection method consults the neighborhood of an object, which is defined by a given radius [10]. An object is then considered an outlier if its neighborhood does not have enough other points. A density-based outlier detection method [11] investigates the density of an object and that of its neighbors. Here, an object is identified as an outlier if its density is relatively much lower than that of its neighbors.

Local Outlier Factor (LOF) is considered as the most popular density-based outlier detection [7, 12]. The main idea of LOF as proposed in [13] is using the relative density of an object against its neighbors. This relative density is used to assign a degree of being an outlier. This degree is called local outlier factor (LOF). Although there are many research efforts in the literature on simplifying, and enhancing LOF algorithm [14, 15, 16], more enhancements need to be done to deal with big data. LOF', LOF'', Grid LOF [17], and FastLOF [18] are examples of these efforts.

This paper proposes a new outlier detection algorithm based on enhancement of LOF algorithm. Despite the effort done to enhance the LOF algorithm, it still suffers from the complexity problem which appears clearly when dealing with big data. To improve the performance of the LOF, the proposed algorithm focuses on simplifying the step of finding the nearest neighbors nodes which is the major bottleneck in this algorithm. To prove the effectiveness of the proposed algorithm, time is chosen as a performance metric to assess the efficiency against the current algorithms.

The remainder of this paper is structured as follows. Section 2 provides related work about the outlier detection problem. A background about LOF algorithm is introduced in section 3. Section 4 presents the details of the proposed algorithm. Section 5 introduces the simulation results and provides an analytical discussion on the performance of the proposed algorithm. Section 6 introduces the conclusions and future work.

2. RELATED WORK

The importance of outlier detection comes from the fact that the deduced data can be translated into actionable information that can be used in many applications. These applications include but not restricted to fraud detection for credit cards, control systems, medical research, communication at runtime software, image sharing [32], intelligent transportation system, wireless sensor networks, and even human skin detection.

An extensive research effort has been seen in the literature tackling outlier detection. In [19], an overview of the existing outlier detection techniques in database management and data mining fields is provided. The introduced techniques have been classified according to different dimensions. Generally, distance-based [10] and density-based [11] outlier detection methods are the most important methods for outlier detection. The Major difference between the two methods is the granularity level [9] as distance-based methods give a higher level of granularity.

Distance-based methods are based on the nearest neighbor distances [9]. A point is considered as an outlier point if it has k-nearest neighbor distances larger than normal points. The detailed granularity level of distance-based methods gives more capacities in outlier analysis but with more computational cost [9]. There are many approaches for outlier detection in this category, such as cell-based approach [9], nested loop approach [3], and reverse nearest neighbor approach [9].

In real-world data sets, the structure of data is more complex as outliers are considered by their local neighborhoods or by the global data distribution [3]. Getting outliers by global data distribution is called density-based outlier detection. Thus outlier is detected by considering the density around each point in the data set. There are many approaches for outlier detection in this category such as Local Correlation Integral (LOCI) [9], histogram-based approaches [9], kernel density estimation approaches [9], and Local Outlier Factor (LOF) [9, 13]. Besides being the most popular density-based outlier detection approach, LOF is one of the simplest approaches for outlier detection [19].

In [13], each object is assigned a degree of being outlier. This degree is called the local outlier factor (LOF) of an object. This degree depends on how isolated the object is with respect to the surrounding neighborhood. LOF algorithm has been used in many applications such as cloud computing [20], network outlier detection [21], data clustering [22], fault analysis and detection [23], and steel plates fault diagnosis [24].

Towards enhancing LOF algorithm, many efforts have been seen in the literature. Kernel Density-Based Local Outlier Factor (KLOF) is an outlier detection algorithm which is based on LOF [25]. In [26], a hierarchical framework using approximated LOF is used for efficient anomaly detection. Also, an enhanced approach for LOF is proposed to be used for data mining purposes in [27]. The complexity of finding the nearest neighbors in LOF algorithm is $O(N^2)$ where the complexity of the algorithm itself is $O(N)$. Thus, many researchers tried to skip the step of finding the nearest neighbors in the LOF algorithm. In this path, LOF', LOF'', and GridLOF have been proposed in [17]. GridLOF is the most efficient and adaptive algorithm in calculating LOF value for each data objects in the data set. GridLOF algorithm also increases accuracy as it avoids some false identification that may occur in LOF. FastLOF has been also proposed in [18] to speed up the LOF computation. This is done by randomly dividing the dataset into groups. For each group, LOF is calculated and the point with LOF value greater than the defined threshold is identified (the threshold is the initially selected between 1.0 and 2.0). This process is repeated to find better neighbors. Although, FastLOF algorithm [18] can get outliers in any dataset, it cannot get all neighbors of a point as the data sets are divided randomly into groups.


3. BACKGROUND

As the proposed work in this paper is based on enhancing the LOF algorithm, some details about LOF and its enhanced version GridLOF are provided in this section. In [13], LOF is presented with the steps shown in figure 1.

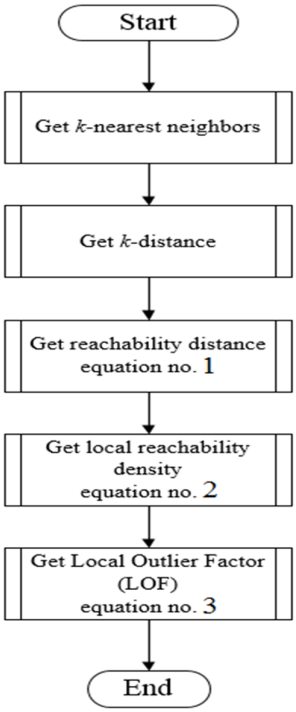


Figure 1: LOF Calculation Steps

1- Get *k*-nearest neighbors for an object *p*.
2- Get *k*-distance for an object *p*.
3- Get reachability distance of an object *p* with his *k*-nearest neighbors:

$$reach - dist_k(p,o) = max\big(k - distance(o), d(p,o)\big) \quad (1)$$

where *d(p, o)* is the distance between object *p* and its neighbor object *o*

4- Get local reachability density of an object p:

$$lrd_{MinPts}(p) = 1 \Big/ \left[\frac{\sum_{o \in N_{MinPts(p)}} reach - dist_{MinPts}(p,o)}{|N_{MinPts}(p)|}\right] \quad (2)$$

This is based on the minimum points (*MinPts*) which is the nearest neighbors of *p*.

5- Get LOF for an object *p* as shown in figure 2 by:

$$LOF_{MinPts}(p) = \left[\frac{\sum_{o \in N_{MinPts}(p)} \frac{lrd_{MinPts}(o)}{lrd_{MinPts}(p)}}{|N_{MinPts}(p)|}\right] \quad (3)$$

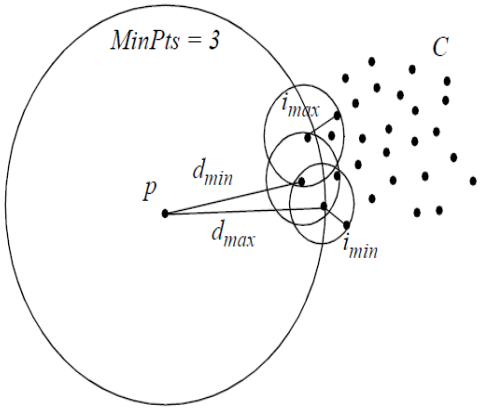


Figure 2: LOF for object *p*

In [17], GridLOF algorithm is proposed as an adaptive algorithm which prunes away the portion of dataset known to be non-outliers. First, each dimension of the data space is quantized into equal width intervals, resulting in a grid-based structure. Then, for each non empty grid cell c, the neighboring grid cells are examined and c is labeled as a boundary cell once a neighboring grid cell with less than or equal to the pre-defined threshold (σ) number of points residing in it is found. σ is a relatively a small number. In the extreme case, σ can be set to zero. Finally, only the LOF values of points inside boundary cells are calculated using above mentioned LOF (5 steps). Figure 3 illustrates the idea of GridLOF algorithm.

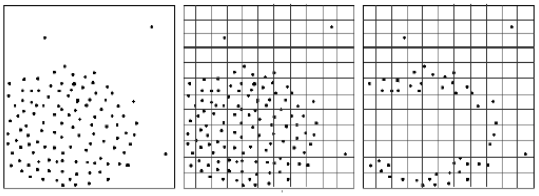


Figure 3: GridLOF algorithm

4. THE PROPOSED APPROACH

In GridLOF algorithm, the portion of dataset known to be non-outliers is prune away [17]. Local outlier factor (LOF) of the remaining points is then calculated. Although the overall cost for computing LOF can be reduced, GridLOF still has complexity of $O(N^2)$.

The proposed algorithm in this paper replaces the first step in the original LOF algorithm, which gets the *k*-nearest neighbors for an object *p*, by another method which gets the nearest neighbors in a more efficient and novel manner. The proposed work is based on re-defining each point P_i by two values: (w, θ) (referred to an index point P_0) instead of the normal coordinates $(x, y)$where:

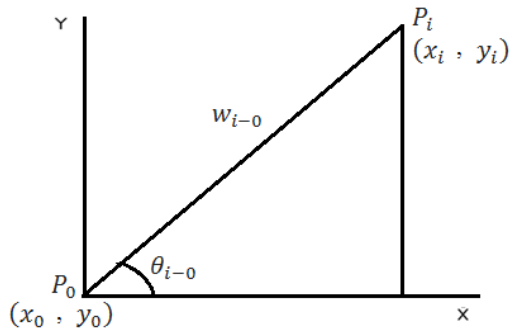


Figure 4: Re-defining each point by (w, θ)

w_{i-0}: is the distance between P_i and P_0 (as shown in figure 4) which is computed as:

$$w_{i-0} = \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2} \quad (4)$$

θ_{i-0}: is the angle between line P_0P_i and X axis which is computed as:

$$\theta_{i-0} = \sin^{-1}\left(\frac{|y_i - y_0|}{w_{i-0}}\right) = \cos^{-1}\left(\frac{|x_i - x_0|}{w_{i-0}}\right) \quad (5)$$

For each point, a circle is drawn with radius R. The equations representing all the points within the circle are deduced which refer only to $W, \theta,$ and R. To get these equations, we need to define the following variables, as shown in figure 5:

φ_{i-0}: is the angle between line P_0P_i and circle's tangent from P_0 which is computed as:

$$\varphi_{i-0} = \sin^{-1}\left(\frac{R}{w_{i-0}}\right) \quad (6)$$

w_{i-0}^-: is the minimum distance between P_0 and the circle which is computed as:

$$w_{i-0}^- = w_{i-0} - R \quad (7)$$

w_{i-0}^+ is the maximum distance between P_0 and the circle which is computed as:

$$w_{i-0}^+ = w_{i-0} + R \quad (8)$$

θ_{i-0}^-: is the minimum angle between circle's tangent from P_0 and X axis which is computed as:

$$\theta_{i-0}^- = \theta_{i-0} - \varphi_{i-0} \quad (9)$$

θ_{i-0}^+: is the maximum angle between circle's tangent from P_0 and X axis which is computed as:

$$\theta_{i-0}^+ = \theta_{i-0} + \varphi_{i-0} \quad (10)$$

Then, the hashed area in figure 5 can be described by the following g two equations:

$$w_{i-0}^- \le w_{n-0} \le w_{i-0}^+ \quad (11)$$

$$\theta_{i-0}^- \le \theta_{n-0} \le \theta_{i-0}^+ \quad (12)$$

Thus, a point P_n lies in the hashed area if it satisfies equations 11&12.

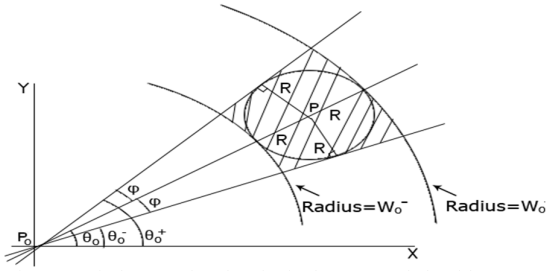


Figure 5: Circle around each point in the proposed algorithm

In order to define the equation of the circle of center P shown in figure 5, we use four index points (P_0, P_1, P_2 and P_3), as shown in figure 6, instead of one point and thus:

$$w_{i-0}^- \le w_{n-0} \le w_{i-0}^+ \quad (13)$$

$$\theta_{i-0}^- \le \theta_{n-0} \le \theta_{i-0}^+ \quad (14)$$

$$w_{i-1}^- \le w_{n-1} \le w_{i-1}^+ \quad (15)$$

$$\theta_{i-1}^- \le \theta_{n-1} \le \theta_{i-1}^+ \quad (16)$$

$$w_{i-2}^- \le w_{n-2} \le w_{i-2}^+ \quad (17)$$

$$\theta_{i-2}^- \le \theta_{n-2} \le \theta_{i-2}^+ \quad (18)$$

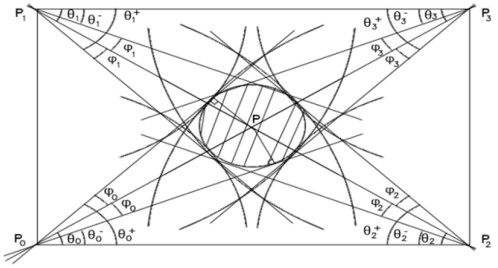


Figure 6: Defining a circle of center P with four index points (P_0, P_1, P_2 and P_3)

$$w_{i-3}^- \le w_{n-3} \le w_{i-3}^+ \quad (19)$$

$$\theta_{i-3}^- \le \theta_{n-3} \le \theta_{i-3}^+ \quad (20)$$

A point P_n lies in the hashed circle if it satisfies equations 13 through 20. To use the above idea in getting the k-nearest neighbor, for each point P in the dataset, a circle with initial radius R is drawn. The number of points in this circle is deduced using the (w, θ) values which are calculated only once with each index point using equations 4 & 5. The radius of this circle is then increased until the number of points in the circle becomes equal to the required k-neighbors.

However, equations 13 through 20 don't represent the circle accurately when we work to get more number of neighbors (the equations represents the hashed shape in figure 5). For that, if we need to get the nearest ten neighbors for example, twelve neighbors are used to insure that we get the required ten neighbors inside the circle not inside the hashed shape. Thus two neighbors are added as an error ratio.

The complexity of the proposed algorithm, in comparison to the GridLOF algorithm, is only $O(N)$ while it is $O(N^2)$ in GridLOF algorithm [17]. In GridLOF, to find the nearest neighbors for a point, the distance between this point and the other points in the dataset is calculated. Thus, to get the nearest neighbors for whole dataset, $N * N$ calculations are performed (where N is the size of the dataset). Whereas in the proposed algorithm, (w, θ) for each point is calculated only once with each index point and the radius R is increased until getting needed neighbors. Thus, only N calculations are done.

5. SIMULATIONS AND RESULTS

Our experiments are performed using Windows application written in C# running on Windows 7 with Intel® Core™ i7-3630QM @ 2.4 GHz processor and

16 GB RAM. Datasets are generated using *R* application [28] with sizes between 60K and 1000K and outlier of 0.1% to 0.5% of the dataset. Table 1 shows the different datasets. Implementation is made with grid size between 0.34*0.34 for 60K datasets, 0.5*0.5 for 125K datasets, 0.75*0.75 for 500K datasets, and 0.9*0.9 for 1000K datasets.

In these experiments, we will examine using the proposed step in the GridLOF algorithm. The main problem in the implementation is how to increase the radius of the circle. We started with initial value for *R* equals 2 and then it is increased by 2 (*R += 2*) for each iteration. Some points (such as outliers) in the dataset need many iterations to get nearest neighbors. Thus we need to limit the number of iterations (10 iterations is chosen) in order to save time and then we switch to the normal GRidLOF algorithm

Table 1 shows a comparison between GridLOF with and without the proposed modification using the execution time as an assessment metric. We have varied the size of the dataset and the number of outliers in it to test the performance at different environments. The last column in the table shows the improvement made by the proposed algorithm compared with the traditional GridLOF algorithm in terms of execution time for all test cases used.

Table 1: execution time results

Dataset			**Execution Time (Seconds)**		**Improvement Percentage (difference) / (GridLOF) %**
Size	**Outlier %**	**K (nearest neighbors)**	**GridLOF**	**Modified GridLOF**	
60K	0.1%	10	240.628763	222.1957088	7.66 %
		15	248.961240	227.5250138	8.61 %
		21	252.651451	232.0782741	8.14 %
	0.2%	10	257.855749	239.8187169	7.00 %
		15	260.793917	245.9040646	5.71 %
		21	267.649309	249.8922932	6.63 %
	0.5%	10	240.403750	228.5240708	4.94 %
		15	244.709997	230.5451863	5.79 %
		21	249.880292	235.0324430	5.94 %
125K	0.1%	10	510.830218	488.4689388	4.38 %
		15	516.922567	491.7331255	4.87 %
		21	524.575004	494.8103017	5.67 %
	0.2%	10	359.346554	339.7514326	5.45 %
		15	362.343725	341.9735596	5.62%
		21	370.563195	348.1349121	6.05 %
	0.5%	10	575.717929	554.9917438	3.60 %
		15	616.376255	562.2411583	8.78 %
		21	647.162016	566.5004021	12.46 %
500K	0.1%	10	1754.697363	1631.817335	7.00%
		15	1815.866862	1635.92757	9.91%
		21	1818.568016	1641.982916	9.71%
	0.2%	10	1902.238802	1798.229853	5.47%
		15	1906.350037	1803.315144	5.40%
		21	1891.903211	1812.294657	4.21%
	0.5%	10	1350.35752	1275.496954	5.54 %
		15	1363.03296	1285.336517	5.70 %
		21	1378.05982	1292.721940	6.19 %
1000K	0.5%	10	2886.046073	2759.082811	4.40%
		15	2976.336237	2783.228192	6.49%
		21	2985.340718	2810.691762	5.85%

Figure 7 shows the execution time (in seconds) for both GridLOF and proposed modified GridLOF in case of data with size equal 60K and outlier percentage ranging from 0.1% to 0.5%. Results show that with increasing the number of nearest neighbors used in calculating LOF value for each point (K), the execution time increases. However, the execution time for the proposed modified GridLOF is still lower than that of normal GridLOF.

Figure 8 shows the execution time (in seconds) for both GridLOF and the proposed modified GridLOF in case of dataset with size equal 125K and outlier percentage ranging from 0.1% to 0.5%. Results show that the proposed modified GridLOF algorithm still outperforms the normal GridLOG algorithm.

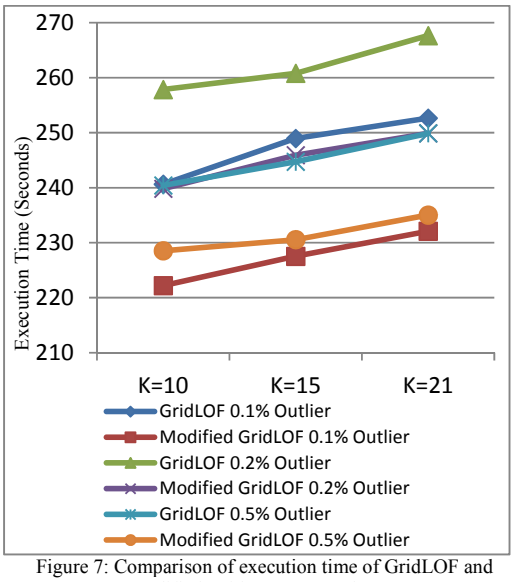



Figure 7: Comparison of execution time of GridLOF and Modified GridLOF at 60K dataset

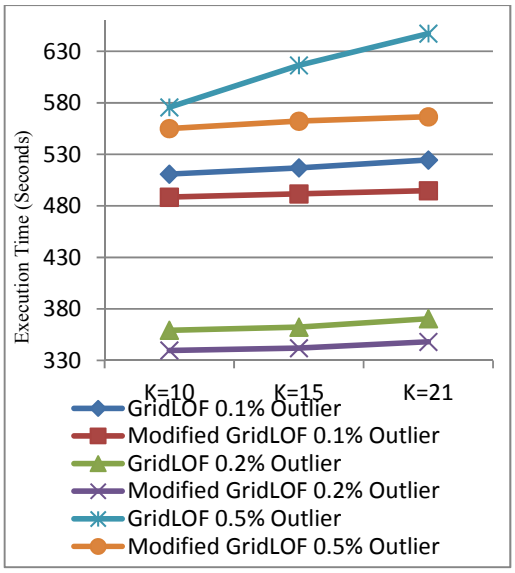


Figure 8: Comparison of execution time of GridLOF and Modified GridLOF at 125K Dataset

Figure 9 shows the superiority of the proposed GridLOF algorithm in case of dataset with size equal 500K and outlier percentage of from 0.1% to 0.5% compared with the normal GridLOF algorithm.

Figure 10 shows the superiority of the proposed GridLOF algorithm in case of dataset with size equal 1000K and outlier percentage 0.5% compared with the normal GridLOF algorithm.

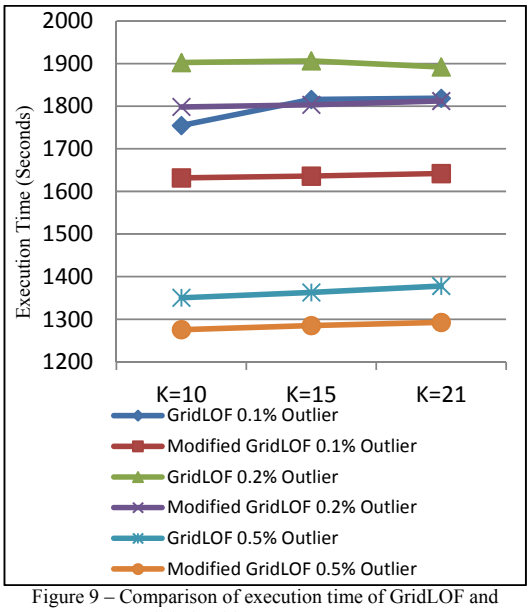


Figure 9 – Comparison of execution time of GridLOF and Modified GridLOF at 500K Dataset

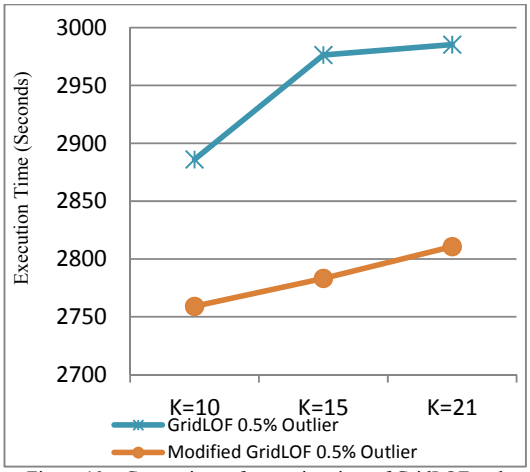


Figure 10 – Comparison of execution time of GridLOF and Modified GridLOF at 1000K Dataset

From Figure 7, 8, 9 & 10, experiments prove that the proposed modified GridLOF algorithm has a better performance than the normal GridLOF in all cases covered in the experiments and it is expected that this will continue in case of increasing data's size, Outlier percentage, or even K (k-nearest neighbors) value.

In order to check the performance of the proposed algorithm in terms of accuracy, new experiments are done. The proposed algorithm is used to detect and remove noise in images using median filter. The steps of applying the proposed algorithm on images are shown in figure 11.

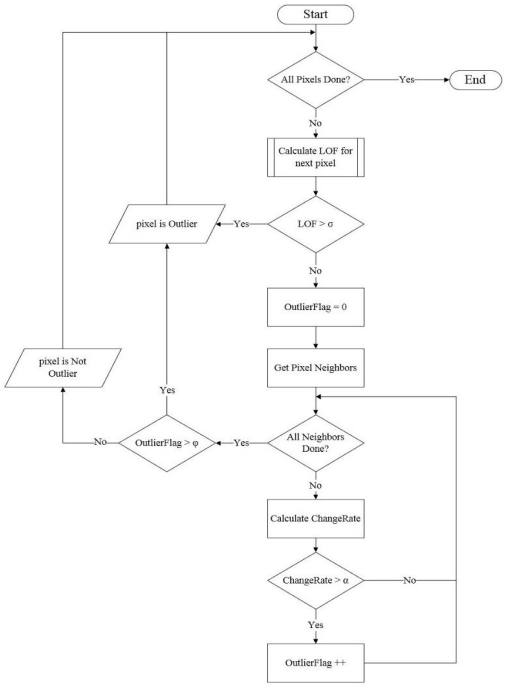


Figure 11: Steps to detect outlier in image

The value of φ depends on needed level of saved useful information in the noisy image. In our experiments $\varphi = 0$ is used. A comparison of the proposed algorithm and the normal median filter [29] using PNSR (Peak signal-to-noise ratio). Figure 12 shows the original used image and the image with 5% noise.

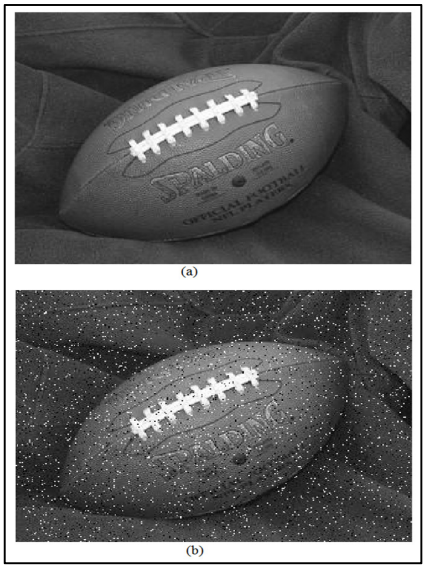


Figure 12: image 1 (a) original one (b) with 5% Salt & Pepper noise

Figure 13 shows the comparison results of filtering the noisy image (with 5% noise level) using the normal median filter and the proposed algorithm. The same experiments are done with 10%, and 20% noise levels and the PSNR is calculated in each case and the results are shown in figure 14.

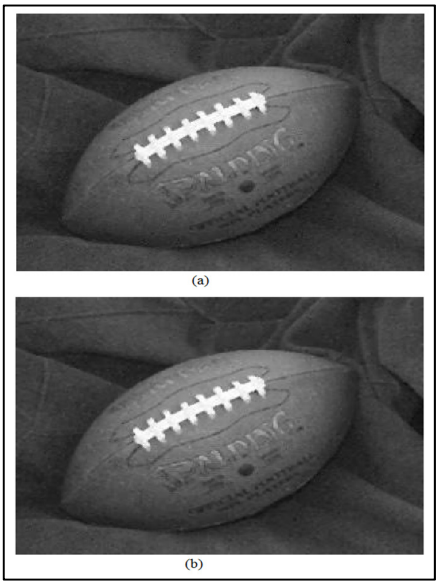


Figure 13: image 1 (5% noise) after apply median filter (a) Proposed algorithm (b) Normal median filter

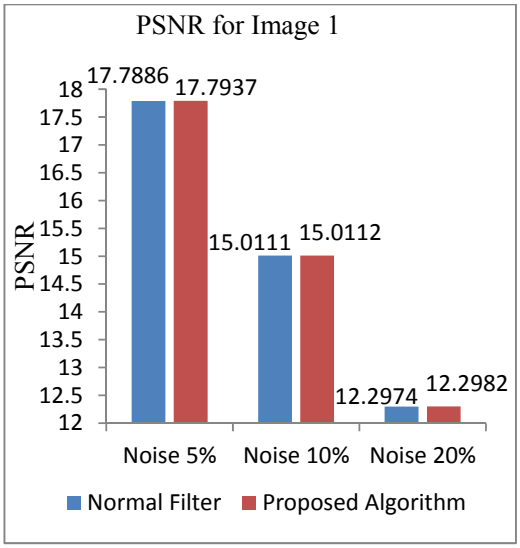


Figure 14: PSNR for image 1 with 5%, 10%, & 20% noise

Figure 14 shows that proposed algorithm has higher PSNR with 5%, 10%, 20% outlier percentages which proves that the proposed modified GridLOF algorithm has higher level of accuracy of detecting outliers in any dataset compared with normal median filter.

6. CONCLUSIONS

Outlier detection can be seen in many applications such as fraud detection for credit cards, control systems, medical research, image sharing, wireless sensor networks, and even human skin detection. This paper proposed a new outlier detection algorithm based on enhancement of LOF algorithm. The proposed algorithm focused on simplifying the step of finding nearest neighbors. Time and accuracy are chosen as performance metric to assess the

efficiency of the proposed algorithm. The proposed algorithm outperforms all kinds of LOF algorithm in terms of speed. The proposed algorithm is also used for image correction by detecting and removing outliers in the image.

Future work will focus on solving some issues that occurred during working with GridLOF algorithm such as increasing the radius of the circle which includes any point and iterations limits before getting the nearest neighbors.

REFERENCES

[1] C. Eaton, D. Deroos, T. Deutsch, G. Lapis, P. Zikopoulos, Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data IBM 2011.

[2] M. Barlow, Real-Time Big Data Analytics 2013.

[3] J. Han, M. Kamber, Data Mining: Concepts and Techniques 2000.

[4] E. Rahm, H. H. Do, Data Cleaning: Problems and Current Approaches, IEEE 2000.

[5] R. Cooley, B. Mobasher, J. Srivastava, Data Preparation for Mining World Wide Web Browsing Patterns, Knowledge and Information Systems, 1, pp, 5–32 1999.

[6] D. M. Hawkins, Identification of Outliers, Chapman and Hall 1980.

[7] R. A. Johnson, Applied Multivariate Statistical Analysis. Prentice Hall 1992.

[8] V. Barnett, T. Lewis, Outliers in Statistical Data. John Wiley 1994.

[9] C. C. Aggarwal, An Introduction to Outlier Analysis 2013.

[10] O. Z. Maimon, L. Rokach, Data Mining and Knowlege Discovery Handbook, Springer 2010.

[11] L. Duan, L. Xu, Y. Liu, J. Lee, Cluster-based outlier detection, Annals of Operations Research (Volume 168, Issue 1 , pp 151-168) Springer 2008.

[12] W. Jin, A. K. H. Tung, J. Han, Mining top-n local outliers in large databases, Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining (Pages 293-298) ACM 2001.

[13] M. M. Breunig, H. Kriegel, R. T. Ng, J. Sander, LOF: Identifying Density-Based Local Outliers, Proc. ACM SIGMOD 2000 Int. Conf. On Management of Data, Dalles, TX 2000.

[14] J. Tang, Z. Chen, A. W Fu, D. W. Cheung, Enhancing Effectiveness of Outlier Detections for Low Density Patterns, Advances in Knowledge Discovery and Data Mining (Volume 2336 of the series Lecture Notes in Computer Science pp 535-548) Springer Berlin Heidelberg 2002.

[15] S. Jiang, Q. Li, K. Li, H. Wang, Z. Meng, GLOF: a new approach for mining local outlier, Machine Learning and Cybernetics, International Conference on (Volume:1), IEEE 2003.

[16] Z. He, X. Xu, S. Deng, Discovering cluster-based local outliers, Pattern Recognition Letters (Volume 24, Issues 9–10) Elsevier Science 2003.

[17] A. L. Chiu, A. W. Fu, Enhancements on Local Outlier Detection, Database Engineering and Applications Symposium, 2003. Proceedings. Seventh International, IEEE 2003.

[18] M. Goldstein, FastLOF: An Expectation-Maximization based Local Outlier Detection Algorithm, 21st International Conference on Pattern Recognition, IEEE 2012.

[19] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey, ACM Computing Surveys 2009.

[20] T. Huang, Y. Zhu, Q. Zhang, Y. Zhu, D. Wang, M. Qiu, L. Liu, An LOF-based Adaptive Anomaly Detection Scheme for Cloud Computing, IEEE 37th Annual Computer Software and Applications Conference Workshops 2013.

[21] E. Schubert, A. Zimek, H. Kriegel, Local outlier detection reconsidered: a generalized view on locality with applications to spatial, video, and network outlier detection, Springer 2012.

[22] V. SALTENIS, Data Clustering Based on Maximization of Outlier Factor, Journal of Global Optimization, Springer 2006.

[23] Y. Zhao, F. Balboni, T. Arnaud, J. Mosesian, R. Ball, B. Lehman, Fault Experiments in a Commercial-Scale PV Laboratory and Fault Detection Using Local Outlier Factor, Photovoltaic Specialist Conference (PVSC) IEEE 40th 2014.

[24] Z. Zhao, J. Yang, W. Lu, X. Wang, Application of Local Outlier Factor method and Back-Propagation Neural Network for steel plates fault diagnosis, Control and Decision Conference (CCDC) 27th IEEE 2015.

[25] M. Dandan, Q. Xiaowei, W. Weidong, Anomalous Cell Detection with Kernel Density-Based Local Outlier Factor, China Communications (Volume:12 Issue: 9), IEEE 2015.

[26] L. Xu, Y. Yeh, Y. Lee, J. Li, A Hierarchical Framework Using Approximated Local Outlier Factor for Efficient Anomaly Detection, The 3rd International Workshop on Sensor Networks for Intelligence Gathering and Monitoring (SNIGM), Procedia Computer Science 2013.

[27] V. Bhatt, K. G. Sharma, A. Ram, An Enhanced Approach for LOF in Data Mining, Proceedings of 2013 International Conference on Green High Performance Computing, IEEE 2013.

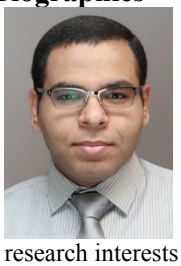
[28] R, data analysis software, more info. http://www.inside-r.org/

[29] I. Pitas, A. N. Venetsanopoulos, Order statistics in digital image processing, Proc. IEEE, vol. 80, no. 12, Decamber 1992.

Tables

Figures

Biographies



Eslam Mahmoud, received the B.Sc degree in Computer Science and Automatic Control from the Faculty of Engineering, Tanta University, in 2009. He is working now as Senior Software Developer. His research interests include; Software Engineering, Database and Big Data.

Ahmed M. Elmogy received his B.Sc., and M.Sc from Computers & Control Dept., Faculty of Eng., Tanta Univ., Egypt in 1998, and 2003. He is awarded his Ph.D. from ECE Dept., Waterloo Univ., Canada. He is currently working as an Assistant Professor, Computers & Control Dept., Faculty of Eng., Tanta Univ., Egypt. His research interests include; Data Mining, Artificial Intelligence, Cryptography, and Robotics.

Amany Sarhan, received the B.Sc degree in Electronics Engineering, and M.Sc. in Computer Engineering from the Faculty of Engineering, Mansoura University, in 1990, and 1997, respectively. She awarded the Ph.D. degree as a joint research between Tanta Univ., Egypt and Univ. of Connecticut, USA. She is working now as a Full Prof. and head Computers and Control Dept., Tanta Univ., Egypt. Her research interests include; Distributed Systems, Software Restructuring, Object-oriented Databases, and Image and video processing, GPU and Distributed Computations.